Adaptive Join Operators for Result Rate Maximization on Multiple Inputs

[1]N.Veerendra Reddy, [2]G.Aparna
[1,2]Dept. of IT, Aditya Engineering College Surampalem, AP, India

Abstract
Adaptive join algorithms have recently attracted a lot of attention in emerging applications where data are provided by autonomous data sources through heterogeneous network environments. Their main advantage over traditional join techniques is that they can start producing join results as soon as the first input tuples are available, thus, improving pipelining by smoothing join result production and by masking source or network delays. In this paper, We are introducing the concept of Split Query Parallel Processing (SPQP) before implementing Double Index NEsted-loops Reactive join (DINER), a new adaptive two-way join algorithm for result rate maximization. DINER combines two key elements: an intuitive flushing policy that aims to increase the productivity of in-memory tuples in producing results during the online phase of the join, and a novel reentrant join technique that allows the algorithm to rapidly switch between processing in-memory and disk-resident tuples, thus, better exploiting temporary delays when new data are not available. We then extend the applicability of the proposed technique for a more challenging setup: handling more than two inputs. Multiple Index NEsted-loop Reactive join (MINER) is a multiway join operator that inherits its principles from DINER. Our experiments using real and synthetic data sets demonstrate that the proposed method outperforms than the previousoptimization techniques Ourexperiments also shows that in the presence of multiple inputs, MINER manages to produce a high percentage of early results, outperforming existing techniques for adaptive multiway join.

Keywords
Query Processing, Join, DINER, MINER, Streams

I. Introduction

MODERN information processing is moving into a realm where we often need to process data that are pushed orpulled from autonomous data sources through heterogeneous networks. Adaptive query processing has emerged as an answer to the problems that arise because of the fluidity and unpredictability of data arrivals in such environments [1]. An important line of research in adaptive query processing has been toward developing join algorithms that can produce tuples "online," from streaming, partially available input relations, or while waiting for one or more inputs [4,7,9,12,14-15]. Such nonblocking join behavior can improve pipelining by smoothing or "masking"varying data arrival rates and can generate join results with high rates, thus, improving performance in a variety ofquery processing scenarios in data integration, onlineaggregation, and approximate query answering systems. Compared to traditional join algorithms (be they sort-, hash-, or nested-loop-based [13]), adaptive joins are designed to deal with some additional challenges: The input relations they use are provided by external network sources. Theimplication is that one has little or no control over the order orrate of arrival of tuples. Since the data source reply speed,streaming rate and streaming order, as well as network traffic and congestion levels, are unpredictable, traditional join algorithms are often unsuitable or inefficient. For example, most traditional join algorithms cannot produce results until at least one of the relations is completely available. Waiting for one relation to arrive completely to produce results is often unacceptable. Moreover, and more importantly, in emerging data integration or online aggregation environments, a key performance metric is rapid availability of first results and a continuous rate of tuple production.

Split based parallel query processing techniques were created often increase the performance of the query result rate with compare to the previous techniques. and in order to lift the limitations of traditional join algorithms in such environments. By being able to produce results whenever input tuples become available, adaptive join algorithms overcome situations like initial delay, slow data delivery, or bursty arrival, which can affect the efficiency of the join. All existing algorithms work in three stages. During the Arriving phase, a newly arrived tuple is stored in memory and it is matched against memory-resident tuples belonging to the other relations participating in the join. Since the allocated memory for the join operation is limited and often much smaller than the volume of the incoming data, this results in tuple migration to disk. The decision on what to flush to disk influences significantly the number of results produced during the Arriving phase. The Arriving phase is suspended when all data sources are temporarily blocked and a Reactive phase kicks in and starts joining part of the tuples that have been flushed to disk. An important desideratum of this phase is the prompt handover to the Arriving phase as soon as any of the data sources restarts sending tuples. Each algorithm has a handover delay which depends on the minimum unit of work that needs to be completed before switching phases. This delay has not received attention in the past, but we show that it can easily lead to input buffer overflow, lost tuples, and hence incorrect results. When all sources complete the data transmission, a Cleanup phase is activated and the tuples that were not joined in the previous phases (due to flushing of tuples to disk) are brought from disk and joined. Even if the overall strategy has been proposed for a multiway join, most existing algorithms are limited to a two-way join. Devising an effective multiway adaptive join operator is a challenge in which little progress has been made [15].

In this paper, we propose two new techniques for query processing, the query may be split into different queries according to the query type, join type and and parallel processing of the splitted quires and follows the adaptive join algorithms for output rate maximization in data processing over autonomous distributed sources.

The first the system accepts query, and estimate the query length and split the query according to the query type, join type etc and the system parallel process the spitted queries by following the Adaptive operator algorithms for output rate maximization in data processing over autonomous distributed sources

The first algorithm, Double Index NEsted-loop Reactive join (DINER) is applicable for two inputs, while Multiple Index NEsted-loop Reactive join (MINER) can be used for joining an arbitrary number of input sources.DINER follows the same overall pattern of executiondiscussed earlier but incorporates a series of noveltechniques and ideas that make it faster, leaner (in memory use), and more adaptive than its predecessors. More specifically,

the key differences between DINER and existing algorithms are [a] The key idea of our flushing policy [a] .MINER extends DINER to multiway joins and it maintainsall the distinctive and efficiency generating properties of DINER. MINER maximizes the output rate by [a]. An important feature of both DINER and MINER is that they are the first adaptive, completely unblocking join techniques that support range join conditions. Range join queries are a very common class of joins in a variety of applications, from traditional business data processing to financial analysis applications and spatial data processing.

Progressive Merge Join (PMJ) [4], one of the early adaptive algorithms, also supports range conditions, but its blocking behavior makes it a poor solution given the requirements of current data integration scenarios.

Our experiments show that our proposed techniques outperformed by DINER.

II. Related Work

Existing work on adaptive join techniques can be classified in two groups: hash based [6-7,9,12,14-15] and sort based [4]. Examples of hash-based algorithms include DPHJ [7] and XJoin [14], the first of a new generation of adaptive nonblocking join algorithms to be proposed. XJoin was inspired by Symmetric Hash Join (SHJ) [6], which represented the first step toward avoiding the blocking behavior of the traditional hash-based algorithms. SHJ required both relations to fit in memory; however, XJoin removes this restriction. MJoin [15], algorithm appeared as an enhancement of XJoin in which its applicability isextended to scenarios where more than two inputs arepresent. The above-mentioned algorithms were proposed for data integration and online aggregation. Pipelined hash join [16], developed concurrently with SHJ, is also an extension of hash join and was proposed for pipelined query plans in parallel main memory environment. Algorithms based on sorting were generally blocking, since the original sort merge join algorithm required an initial sorting on both relations before the results could be obtained. Although there were some improvements that attenuate the blocking effect [10], the first efficient nonblockingsort-based algorithm was PMJ [4]. Hash Merge Join (HMJ) [9], based on XJoin and PMJ, is a nonblocking algorithm which tries to combine the best parts of its predecessors while avoiding their shortcomings.

Finally, Rate-based Progressive Join (RPJ) [12] is an improved version of HMJ that is the first algorithm to make decisions, e.g., about flushing to disk, based on the characteristics of the data. In what follows, we describe the main existing techniques for adaptive join. For all hash-based algorithms, we assume that each relation Ri, i ¼A;B is organized in npart buckets. The presentation is roughly chronological. XJoin. As with "traditional" hash-based algorithms, XJoin organizes each input relation in an equal number of memory and disk partitions or buckets, based on a hash function applied on the join attribute. The XJoin algorithm operates in three phases. During the first, arriving, phase, which runs for as long as either of thedata sources sends tuples, the algorithm joins the tuples from the memory partitions. Each incoming tuple is stored in its corresponding bucket and is joined with the matching tuples from the opposite relation. When memory gets exhausted, the partition with the greatest number of tuples is flushed to disk. The tuples belonging to the bucket with the same designation in the opposite relation remain on disk. When both data sources are blocked, the first phase pauses and the second, reactive, phase begins. The last, cleanup, phase starts when all tuples from both data sourceshave completely arrived. It joins the matching tuples that were missed during the previous two phases.

In XJoin, the reactive stage can run multiple times for the same partition. The algorithm applies a time-stamp-based duplicate avoidance strategy in order to detect already joined tuple pairs during subsequent executions. MJoin.MJoin [15], is a hash-based algorithm which extends XJoin and it was designed for dealing with star queries of the form[a]. The algorithm creates as many in-memoryhash tables as inputs. A new tuple is inserted in the hash table associated with its source and it is used to probe the remaining hash tables. The probing sequence is based on the heuristic that the most selective joins should be evaluated first in order to minimize the size of intermediate results. When the memory gets exhausted, the coordinated flushing policy is applied. A hashed bucket is picked (as incase of XJoin) and its content is evicted from each input. MJoin cannot be used directly for joins with multiple attributes or nonstar joins. Viglas et al. [15], describes a naïve extension where only a flushing policy that randomly evicts the memory content can be applied.Progressive Merge Join[a], Hash Merge Join. HMJ [9], Rate-based Progressive Join. RPJ [12], In case both relations are temporarily blocked, RPJ begins its reactive phase, which "combines" the XJoin and HMJ reactive phases. The tuples from one of the disk buckets of either relation can join with the corresponding memory bucket of the opposite relation, as in case of XJoin, or two pairs of disk buckets can be brought in memory and joined as in case of HMJ (and PMJ). The algorithm chooses the task that has the highest output rate. During its cleanup phase, RPJ joins the disk buckets. The duplicate avoidance strategy is similar with the one applied by XJoin.

A. Previous Adoptive Operator Techniques

1. DINER

DINER algorithm for computing the join result of two finite relations RA and RB, which may be stored at potentially different sites and are streamed to our local system. Given the unpredictable behavior of the network, delays and random temporary suspensions in data transmission may be experienced. The goal of DINER is twofold. It first seeks to correctly produce the join result, by quickly processing arriving tuples, while avoiding operations that may jeopardize the correctness of the output because of memory overflow. Moreover, in the spirit of prior work [5,9,12], DINER seeks to increase the number of join tuples (or, equivalently, the rate of produced results) generated during the online phase of the join, i.e., during the (potentially long) time it takes for the input relations to be streamed to our system. To achieve these goals, DINER is highly adaptive to the (often changing) value distributions of the relations, as well as to potential network delays.

2. MINER

the DINER algorithm,termed as MINER, for joining multiple relations. For ease of presentation, we conduct our discussion for joins of the form [a]. We then present the required extensions for other categories of joins. Due to its inherent similarity with DINER, we focus our discussion to the adjustments, compared to DINER, required for the proper operation of MINER. Extending the Online Phase [a], Extending the Reactive Phase [a].

III. Proposed Model

The proposed system (SPQP)specifies two different techniques for query processing before implementing the previous algorithms. The system takes the input query and splits the overall query into

different partitions called split ,then process parallel the each split by follows the previous adaptive process algorithms which enhance the optimization of the query result rate.

Query length estimation method: The system accepts the query astheinput source and the SQL engine performs the specific calculations for the query length based on the query type andfurther used for query splitting according to the estimation result.

Parallel Query Processing: The system takes the advantage from the splitting technique and the system further process the splitted queries parallel by incorporating the adoptive join operator technique for each query and the system works towards query result optimization.

The contributions of our paper are:

1. We introduced two techniques ofor the query optimization query split and parallel query processing.
2. We follows DINER a novel adaptive join algorithm that supports both equality and range join predicates. DINER builds on an intuitive flushing policy that aims at maximizing the productivity of tuples that are kept in memory.
3. DINER is the first algorithm to address the need to quickly respond to bursts of arriving data during the Reactive phase. We propose a novel extension to nested loops join for processing disk-resident tuples when both sources block, while being able to swiftly respond to new data arrivals.
4. We introduce MINER, a novel adaptive multiway join algorithm that maximizes the output rate, designed for dealing with cases where data are held by multiple remote sources.
5. We provide a thorough discussion of existing algorithms, including identifying some important limitations, such as increased memory consumption because of their inability to quickly switch to the Arriving phase and not being responsive enough when value distributions change.
6. We provide an extensive experimental study of DINER including performance comparisons to existing adaptive join algorithms and a sensitivity analysis. Our results demonstrate the superiority of DINER in a variety of realistic scenarios. During the online phase of the algorithm, DINER manages to produce up to three times more results compared to previous techniques. The performance gains of DINER are realized when using both real and synthetic data and are increased when fewer resources (memory) are given to the algorithm. We also evaluate the performance of MINER, and we show that it is still possible to obtain early a large percentage of results even in more elaborated setups where data are provided through multiple inputs.

IV. Experiments

In this section, we present an extensive experimental study of our proposed method. The objective of this study is to enhance the optimization performance. We first evaluate the performance of Proposed (SPQP)against the state of the artDINER [a], for a variety of both real-life and synthetic data sets.we demonstrate SPQP's superior performance over a variety of real and synthetic data sets in an environment without network congestion or unexpected source delays. The following figure shows the overall comparison differences between SPQP and DINNER.

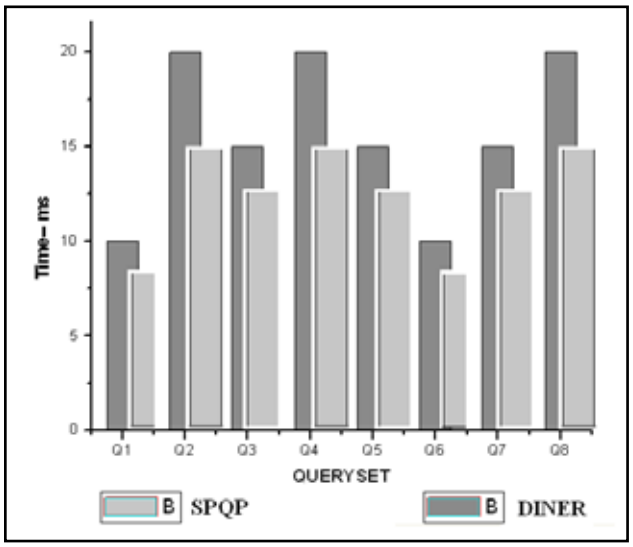

Fig. 1 Overall Comparison Between SPQP and DINER

V. Conclusions and Future Work

In this work, we introduce Split based parallel query processing to enhances the optimization of the result rate and DINER, a new adaptive join algorithm for maximizing the output rate of tuples, when two relations are being streamed to and joined at a local site. The advantages of DINER stem from 1) its intuitive flushing policy that maximizes the overlap among the join attribute values between the two relations, while flushing to disk tuples that do not contribute to the result and 2) a novel reentrant algorithm for joining disk-resident tuples that were previously flushed to disk. Moreover, DINER can efficiently handle join predicates with range conditions, a feature unique to our technique. We also present a significant extension to our framework in order to handle multiple inputs. The resulting algorithm, MINER addresses additional challenges, such as determining the proper order in which to probe the in-memory tuples of the relations, and a more complicated bookkeeping process during the Reactive phase of the join. Through our experimental evaluation, we have demonstrated the advantages of both algorithms on a variety of real and synthetic data sets, their resilience in the presence of varied data and network characteristics and their robustness to parameter changes.

References

[1] S. Babu, P. Bizarro,"Adaptive Query Processing in the Looking Glass", Proc. Conf. Innovative Data Systems Research (CIDR), 2005.

[2] D. Baskins Judy Arrays, [Online] Available: http://www.judy.sourceforge.net, 2004.

[3] M.A. Bornea, V. Vassalos, Y. Kotidis, A. Deligiannakis, "Double Index Nested-Loop Reactive Join for Result Rate Optimization", Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2009.

[4] J. Dittrich, B. Seeger, D. Taylor,"Progressive Merge Join: A Generic and Non-Blocking Sort-Based Join Algorithm", Proc. Int'l Conf. Very Large Data Bases (VLDB), 2002.

[5] P.J. Haas, J.M. Hellerstein,"Ripple Joins for Online Aggregation", Proc. ACM SIGMOD, 1999.

[6] W. Hong, M. Stonebraker,"Optimization of Parallel Query Execution Plans in XPRS", Proc. Int'l Conf. Parallel and Distributed Information Systems (PDIS), 1991.

[7] Z.G. Ives et al.,"An Adaptive Query Execution System for Data Integration", Proc. ACM SIGMOD, 1999.

[8] F. Li, C. Chang, G. Kollios, A. Bestavros,"Characterizing and Exploiting Reference Locality in Data Stream Applications", Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2006.
[9] M.F. Mokbel, M. Lu, W.G. Aref,"Hash-Merge Join: A Non-Blocking Join Algorithm for Producing Fast and Early Join Results", Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2004.
[10] M. Negri, G. Pelagatti,"Join During Merge: An Improved Sort Based Algorithm", Information Processing Letters, Vol. 21, No. 1, pp. 11-16, 1985.
[11] A.S. Tanenbaum,"Modern Operating Systems", Prentice Hall PTR, 2001.
[12] Y. Tao, M.L. Yiu, D. Papadias, M. Hadjieleftheriou, N. Mamoulis,"RPJ: Producing Fast Join Results on Streams through Rate-Based Optimization", Proc. ACM SIGMOD, 2005.
[13] J.D. Ullman, H. Garcia-Molina, J. Widom,"Database Systems: The Complete Book", Prentice Hall, 2001.
[14] T. Urhan, M.J. Franklin,"XJoin: A Reactively-Scheduled Pipelined Join Operator", IEEE Data Eng. Bull., Vol. 23, No. 2, pp. 27-33, 2000.
[15] S.D. Viglas, J.F. Naughton, J. Burger,"Maximizing the Output Rate of Multi-Way Join Queries over Streaming Information Sources", Proc. 29th Int'l Conf. Very Large Data Bases (VLDB '03), pp. 285-296, 2003.
[16] A.N. Wilschut, P.M.G. Apers,"Dataflow Query Execution in a Parallel Main-Memory Environment", Distributed and Parallel Databases, Vol. 1, No. 1, pp. 103-128, 1993.

Assistant.Prof. Mrs. G.Aparna, Working as Asst.Prof in the Department of Information Technology in Aditya Engineering College, Surampalem She Receved M.tech from AndraUnversity In Artificial Intelligence and Robotics she Have 4 Years experience in various Engineering Colleges she designed and guided Many Projects.

Mr. N.veerendra Kumar Reddy is a student of Aditya Engineering college, Surampalem he is pursuing his M.TECH from this college and He received his graduation from Jntu University In the year 2007.